





Python em IA

Bem-vindo ao nosso estudo sobre a linguagem Python em Inteligência Artificial. Este estudo ajudará você a desenvolver programas na linguagem Python para resolver problemas com ênfase em Inteligência Artificial. Vamos falar de mais um algoritmo importante de mineração de dados, o algoritmo K-Means. Bons estudos!

Algoritmos em IA

Existem vários algoritmos para resolver problemas em Inteligência Artificial. A linguagem Python é a linguagem preferencial para desenvolver esses algoritmos porque tem pacotes prontos que contêm algoritmos já implementados para resolver muitos problemas de Inteligência Artificial. Vamos entender como funciona o algoritmo de clusterização chamado K-means.

Algoritmo K-Means

O algoritmo K-Means (chamado também de K-Médias) fornece uma classificação de informações de acordo com os próprios dados. Este tipo de classificação tem como objetivo analisar e comparar os valores numéricos dos dados. Desta maneira, o algoritmo automaticamente vai fornecer uma classificação automática sem precisar de nenhuma supervisão humana, isto é, sem nenhuma pré-classificação existente. Por causa desta característica, o K-Means é considerado como um algoritmo de mineração de dados não supervisionado. Portanto, K-Means é um algoritmo de clusterização (ou agrupamento) que também está disponível na biblioteca Scikit-Learn, da mesma forma que o algoritmo KNN. É considerado um algoritmo de aprendizado não supervisionado (quer dizer, não precisa de entradas de confirmação externas). Ele avalia e agrupa os dados de acor com suas características. Exemplos de agrupamento: clientes com produ

semelhantes, clientes com desejos parecidos, filmes com faixa etária igual, pacientes com sintomas semelhantes etc.



Exemplo prático de implementação do algoritmo K-Means em Python

Vamos fazer um exemplo de implementação em Python. Na linguagem Python, vamos utilizar o pacote scikit-Learn, que possui o K-Means implementado. A única coisa que precisamos fazer é utilizar os métodos e passar os parâmetros da forma certa. Como vamos utilizar esse pacote, precisamos importá-lo porque ele não vem com a instalação padrão do Python, então o ideal é utilizar o Jupyter Notebook, que já vem com os pacotes importantes para ciência de dados. Se não quiser instalar o Anaconda para usar o Jupyter Notebook, podemos utilizá-lo de modo online pelo Google Colab. Eu sugiro ao leitor que utilize o Google Colab, que é bem simples e fácil de utilizar, sem precisar se preocupar em importar ou instalar pacotes.

Por exemplo, vamos supor que existe uma rede de lojas com filiais em todo o Brasil. A diretoria dessa rede quer saber quais são os melhores lugares para montar depósitos de abastecimento (pontos de distribuição), para que a sua logística fique otimizada.

Os passos do algoritmo K-Means para resolver o problema: (Grus, 2016)



Definir um número de agrupamentos, chamado de 'K'.



Definir de forma aleatória um centroide para cada agrupamento (ou cluster).



Calcular, para cada ponto, o centroide de menor distância, onde cada ponto deve pertencer ao centroide mais próximo.



Reposicionar o centroide. A nova posição do centroide deve ser calculada pela média da posição de todos os pontos do agrupamento.

- Repetir os passos 3 e 4 de forma iterativa até obter a melhor posição dos centroides.

Implementação do problema:

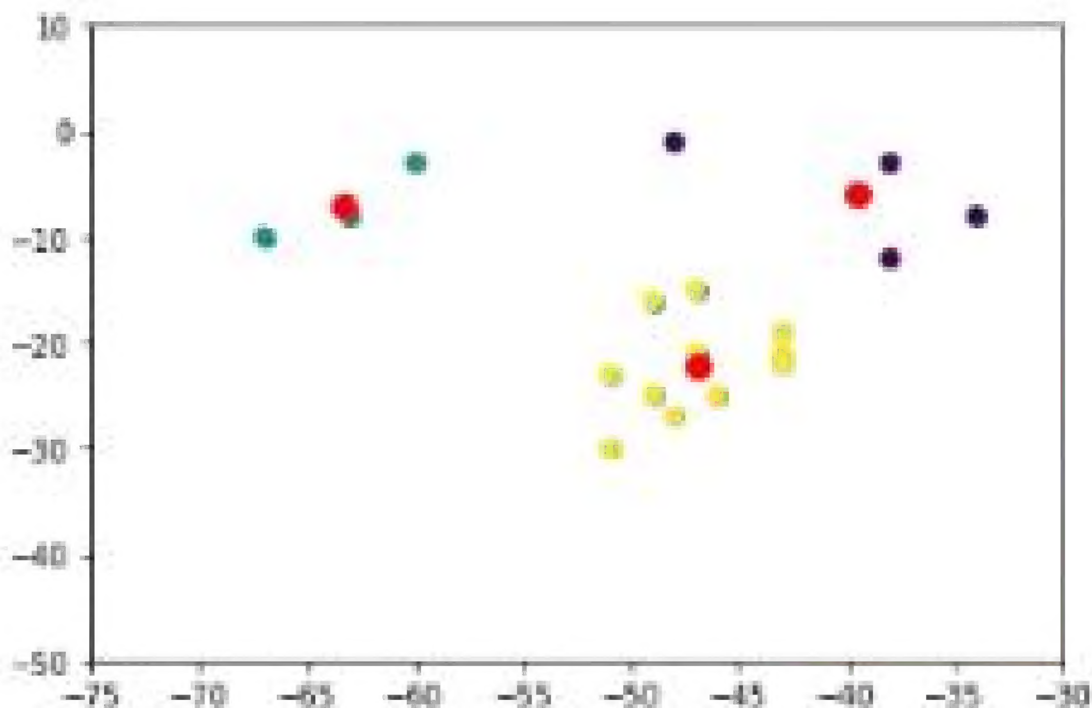
```
import numpy as np #para manipular os vetores
from matplotlib import pyplot as plt #para plotar os gráficos
from sklearn.cluster import KMeans #para usar o KMeans
dataset = np.array(
#matriz com as coordenadas geográficas de cada loja
[[-25, -46], #são paulo
[-22, -43], #rio de janeiro
[-25, -49], #curitiba
[-30, -51], #porto alegre
[-19, -43], #belo horizonte
[-15, -47], #brasilia
[-12, -38], #salvador
[-8, -34], #recife
[-16, -49], #goiania
[-3, -60], #manaus
[-22, -47], #campinas
[-3, -38], #fortaleza
[-21, -47], #ribeirão preto
[-23, -51], #maringa
[-27, -48], #florianópolis
[-21, -43], #juiz de fora
[-1, -48], #belém
[-10, -67], #rio branco
[-8, -63]]) #porto velho
plt.scatter(dataset[:,1], dataset[:,0]) #posicionamento dos eixos x e y
plt.xlim(-75, -30) #range do eixo x
plt.ylim(-50, 10) #range do eixo y
plt.grid() #função que desenha a grade no nosso gráfico
kmeans = KMeans(n_clusters = 3, #numero de clusters
init = 'k-means++', n_init = 10, #algoritmo que define a posição dos clusters de maneira mais assertiva
max_iter = 300) #numero máximo de iterações
pred_y = kmeans.fit_predict(dataset)
plt.scatter(dataset[:,1], dataset[:,0], c = pred_y) #posicionamento dos eixos x e y
plt.xlim(-75, -30) #range do eixo x
plt.ylim(-50, 10) #range do eixo y
plt.grid() #função que desenha a grade no nosso gráfico
plt.scatter(kmeans.cluster_centers_[:,1],kmeans.cluster_centers_[:,0], s = 70, c = 'red') #posição de cada centroide no gráfico
plt.show()
```

Veja que cada linha de código está comentada para melhor compreender o algoritmo. Primeiramente foram importados os pacotes para manipular vetores (numpy), para plotar os gráficos (matplotlib) e para usar o K-Means (sklearn.cluster). Depois foram colocadas em uma matriz as coordenadas geográficas onde se localizam as lojas pertencentes à rede. Em seguida foi definido o posicionamento dos eixos x e y para plotar o gráfico. A seguir é utilizado o algoritmo K-Means para definir a posição dos grupos, com número máximo de 300 iterações. Em seguida é construído o gráfico com o resultado

mostrando cada centroide.



Gráfico do resultado:



Essa clusterização apontou três posições para os depósitos da rede (os pontos vermelhos no gráfico), em Humaitá-AM (posição -7, -63.33333), Acopiara-CE (posição -6, -39.5) e Mogi Guaçu-SP (posição -22.166666, -47).

Atividade Extra

Para se aprofundar no assunto desta aula, pesquise mais sobre o algoritmo K-Means e implemente outros exemplos em Python. Uma opção de pesquisa é o livro Data Science do Zero, de Joel Grus.

Referência Bibliográfica



GRUS, Joel. **Data Science do Zero:** Primeiras regras com o Python. Rio de Janeiro: Alta Books, 2016.

IZBICKI, Rafael, SANTOS, Tiago M. **Aprendizado de Máquina:** Uma abordagem estatística. São Carlos: Rafael Izbicki, 2020.

Ir para exercício